Project Update: A New Foundation of Attack Trees in Linear Logic

Harley Eades III

Computer Science
Augusta University
harley.eades@gmail.com

Abstract. In this short paper I provide an update on the status of newly funded research project investigating founding attack trees in the resource conscious theory called linear logic.

1 Introduction

What is a mathematical model of attack trees? There have been numerous proposed answers to this question. Some examples are propositional logic, multisets, directed acyclic graphs, source sink graphs (or parallel-series pomsets), Petri nets, and Markov processes. Is there a unifying foundation in common to each of these proposed models? Furthermore, can this unifying foundation be used to further the field of attack trees and build new tools for conducting threat analysis?

The answer to the first question is positive, but the answer to the second question is open. Each of the proposed models listed above have something in common. They can all be modeled in some form of a symmetric monoidal category[1] [6,1,2,3] – for the definition of a symmetric monoidal category see Appendix A. That is all well and good, but what can we gain from monoidal categories?

Monoidal categories are a mathematical model of linear logic as observed through the beautiful Curry-Howard-Lambek correspondence [5]. In linear logic every hypothesis must be used exactly once, and hence, if we view a hypothesis as a resource, then this property can be stated as every resource must be consumed. This linearity property is achieved by removing the structural rules for weakening and contraction from classical or intuitionistic logic. Thus, from a resource perspective, resources cannot be spontaneously created or duplicated – hence, propositional logic can be viewed as a degenerate, from a resource perspective, form of linear logic.

Multisets and Petri nets both capture the idea that the nodes of an attack tree capture both the attack action and the state – the resource – of the system being analyzed. As it turns out, linear logic has been shown to be a logical foundation for multisets [6] and Petri Nets [1]. Thus, linear logic has the ability

[1] I provide a proof that the category of source sink graphs is monoidal in Appendix B.

to model the state as well as attack actions of the goals of an attack tree. We propose that linear logic be used as the logical foundation of attack trees.

This projects[2] main goal is to determine the suitability of linear logic as a foundation for attack trees. The type of attack trees we consider in this paper are attack trees with sequential composition similar to Jhawar et al. [4]. Attack trees consist of two layers: the logical layer or process tree, and the quantitative layer. One interesting aspect of our proposed foundation is that the logical layer can come to the aid of the quantitative layer.

At the logical layer the base attacks of an attack tree will correspond to atomic formulas in linear logic, and each branching node of an attack tree will correspond to a binary operator in linear logic. Adding costs corresponds to annotating the atomic formulas with some base cost, and then annotating the binary operators with a cost computed from the costs of their respective left operand (left subtree) and right operand (right subtree). The most interesting aspect of adding costs is that computing the costs at the branching nodes is done during the construction of a derivation using the inference rules of the logic.

The inference rules of linear logic provide a number of benefits when constructing attack trees. First, the inference rules will certify that an attack tree is constructed correctly and all costs on branching nodes will be computed from the costs of the left and right subtrees. On top of that the inference rules offer a means of proving when two attack trees are equivalent. Leveraging the fact that language of attack trees corresponds to a very simple fragment of linear logic, e.g. linear implication is not needed, I conjecture that proving equivalence of attack trees can be automated. Thus, this could be used in practice to certify the correctness of transformations of attack trees maintaining the resource based semantics.

2 Attack Trees without Costs

In this section I introduce attack trees with sequential composition – sometimes referred to SAND attack trees – without cost annotations in the hope that doing so will make the main ideas of this work more clear to the reader. This formulation of attack trees was first proposed by Jhawar et al. [4]. First, I define the syntax of attack trees then a linear logic for reasoning about attack trees.

Definition 1. *Suppose* B *is a set of base attacks whose elements are denoted by* b*. Then an* ***attack tree*** *is defined by the following grammar:*

$$T ::= b \mid T_1 \odot T_2 \mid T_1 \sqcup T_2 \mid T_1 \rhd T_2$$

I denote unsynchronized parallel composition of attacks by $T_1 \odot T_2$*, choice between attacks by* $T_1 \sqcup T_2$*, and sequential composition of attacks by* $T_1 \rhd T_2$*.*

[2] This material is based upon work supported by the National Science Foundation CRII CISE Research Initiation grant, "CRII:SHF: A New Foundation for Attack Trees Based on Monoidal Categories", under Grant No. 1565557.

Now we define the attack tree linear logic (ATLL) for reasoning about attack trees. In ATLL attack trees are modeled as formulas, and thus, if attack trees are formulas, then reasoning about attack trees should correspond to proving implications between attack trees. Furthermore, some care must be taken so that choice and parallel composition are symmetric, but sequential composition is not. We enforce this property in ATLL by separating hypotheses as is usually done in ordered linear logic [**?**].

Definition 2. *The syntax of ATLL formulas and contexts are defined as follows:*

$$\begin{aligned} \textit{(formulas)}\ E &::= b \mid E_1 \odot E_1 \mid E_1 \sqcup E_2 \mid E_1 \rhd E_2 \mid E \multimap B \\ \textit{(base contexts)}\ \Gamma, \Delta &::= \cdot \mid b \mid \Gamma, \Delta \\ \textit{(general contexts)}\ \Theta, \Psi &::= \cdot \mid E \mid \Gamma, \Delta \end{aligned}$$

$$\frac{}{\cdot; b \vdash^T b}\,\mathrm{id}_{\odot} \qquad \frac{}{b; \cdot \vdash^T b}\,\mathrm{id}_{\rhd} \qquad \frac{\Gamma_1; \Delta_1 \vdash^T T_1 \quad \Gamma_2; \Delta_2 \vdash^T T_2}{\Gamma_1, \Gamma_2; \Delta_1, \Delta_2 \vdash^T T_1 \odot T_2}\,\odot$$

$$\frac{\Gamma_1; \Delta_1 \vdash^T T_1 \quad \Gamma_2; \Delta_2 \vdash^T T_2}{\Gamma_1, \Gamma_2; \Delta_1, \Delta_2 \vdash^T T_1 \rhd T_2}\,\rhd \qquad \frac{\Gamma_1; \Delta_1 \vdash^T T_1 \quad \Gamma_2; \Delta_2 \vdash^T T_2}{\Gamma_1, \Gamma_2; \Delta_1, \Delta_2 \vdash^T T_1 \sqcup T_2}\,\sqcup$$

Fig. 1. Well Formed Attack Trees in ATLL

ATLL formulas are defined as an extension of the syntax for attack trees with linear implication. An attack tree then corresponds to an ATLL formula without implication. Suppose E is an ATLL formula such that every base attack in E which is a leaf of sequential composition is in the base context Γ, and every leaf of parallel composition is in the base context Δ. Then E is an attack tree if and only if $\Gamma; \Delta \vdash^T E$ holds with respect to the rules in Figure 1.

The inference rules in Figure 1 enforce several properties. First, they enforce that every base attack is used exactly once, and that an attack tree does not mention linear implication. Furthermore, they allow us to restrict our reasoning to attack trees which I conjecture will result in our reasoning being more amenable to automation.

The ATLL logical inference rules are defined in Figure 2. These rules make up a basic linear logic for attack trees. For example, we can prove that sequential and parallel composition are associative, and the latter is symmetric, but there are a number of facts that these rules do not prove. They cannot prove that choice is associative and symmetric, and they cannot prove the usual distributive laws associated with attack trees with sequential composition [4]. To overcome this limitation we add the attack tree axioms in Figure 3 to ATLL. The logical connective $\circ\!\!-\!\!\circ$ is linear biimplication which is defined in the usual way as classical

$$\frac{}{\cdot; E \vdash E}\,\mathrm{id}_{\odot} \qquad \frac{}{E; \cdot \vdash E}\,\mathrm{id}_{\triangleright} \qquad \frac{\Theta_1; \Psi_1 \vdash E_1 \quad \Theta_2; \Psi_2 \vdash E_2}{\Theta_1, \Theta_2; \Psi_1, \Psi_2 \vdash E_1 \odot E_2}\,\odot_I$$

$$\frac{\Theta_1; \Psi_2 \vdash E_1 \odot E_2 \quad \Theta_2; \Psi_1, E_1, E_2, \Psi_3 \vdash E_3}{\Theta_1, \Theta_2; \Psi_1, \Psi_2, \Psi_3 \vdash E_3}\,\odot_E \qquad \frac{\Theta_1; \Psi_1 \vdash E_1 \quad \Theta_2; \Psi_2 \vdash E_2}{\Theta_1, \Theta_2; \Psi_1, \Psi_2 \vdash E_1 \rhd E_2}\,\rhd_I$$

$$\frac{\Theta_2; \Psi_2 \vdash E_1 \rhd E_2 \quad \Theta_1, E_1, E_2, \Theta_3; \Psi_2 \vdash E_3}{\Theta_1, \Theta_2, \Theta_3; \Psi_1, \Psi_2 \vdash E_3}\,\rhd_E \qquad \frac{\Theta; \Psi_1, E_1, E_2, \Psi_2 \vdash E}{\Theta; \Psi_1, E_2, E_1, \Psi_2 \vdash E}\,\mathrm{sym}_{\odot}$$

$$\frac{\Theta_1; \Psi_1 \vdash E_1 \quad \Theta_2; \Psi_2 \vdash E_2}{\Theta_1, \Theta_2; \Psi_1, \Psi_2 \vdash E_1 \sqcup E_2}\,\sqcup \qquad \frac{\Theta; \Psi, E_1 \vdash E_2}{\Theta; \Psi \vdash E_1 \multimap E_2}\,\multimap_I$$

$$\frac{\Theta_1; \Psi_1 \vdash E_1 \multimap E_2 \quad \Theta_2; \Psi_2 \vdash E_1}{\Theta_1, \Theta_2; \Psi_1, \Psi_2 \vdash E_2}\,\multimap_E$$

Fig. 2. ATLL Logical Inference Rules

$$\frac{\Theta; \Psi \vdash T}{\Theta; \Psi \vdash (T \sqcup T) \circ\!\!\multimap T}\,\mathrm{cont}_{\sqcup} \qquad \frac{\Theta; \Psi \vdash T_1 \sqcup T_2}{\Theta; \Psi \vdash (T_1 \sqcup T_2) \circ\!\!\multimap (T_2 \sqcup T_1)}\,\mathrm{sym}_{\sqcup}$$

$$\frac{\Theta; \Psi \vdash (T_1 \sqcup T_2) \sqcup T_3}{\Theta; \Psi \vdash ((T_1 \sqcup T_2) \sqcup T_3) \circ\!\!\multimap (T_1 \sqcup (T_2 \sqcup T_3))}\,\mathrm{assoc}_{\sqcup}$$

$$\frac{\Theta; \Psi \vdash^{T} T_1 \odot (T_2 \rhd T_3)}{\Theta; \Psi \vdash (T_1 \odot (T_2 \sqcup T_3)) \circ\!\!\multimap ((T_1 \odot T_2) \sqcup (T_1 \odot T_3))}\,\mathrm{dis}_{\odot}$$

$$\frac{\Theta; \Psi \vdash^{T} T_1 \rhd (T_2 \sqcup T_3)}{\Theta; \Psi \vdash (T_1 \rhd (T_2 \sqcup T_3)) \circ\!\!\multimap ((T_1 \rhd T_2) \sqcup (T_1 \rhd T_3))}\,\mathrm{dis}_{\rhd}$$

Fig. 3. ATLL Attack Tree Axioms

implication. The reader should note that these axioms are restricted to attack trees only and not applicable to full ATLL formulas.

The ATLL logical inference rules also make it clear why we separate hypotheses into two contexts. The exchange rule, $\mathrm{sym}_{\odot}$, allows one to exchange hypotheses in the context Ψ, but notice that there is no exchange rule for the context Θ. Thus, one should think of the hypotheses in Ψ as running in parallel and the hypotheses in Θ as running sequentially. In fact, this interpretation is reinforced by the elimination rules, $\odot_E$ and $\rhd_E$, for parallel and sequential conjunction respectively.

Finally, ATLL allows one to prove every equivalence for attack trees with sequential composition given by Jhawar et al. [4] as linear implications. I conjecture that it should be possible to define a proof search algorithm for ATLL to automate equational reasoning on attack trees, but this is future work. However, using implication as a means to reason about attack trees opens the door for different types of reasoning.

Suppose T_1 is a larger attack tree, and we wish to know if the attack tree T_2 is contained within T_1, that is, T_2 is a subattack tree of T_1. Currently, ATLL will not allow us to prove this, but it can if we add the following rules:

$$\frac{\Theta_1, \Theta_2; \Psi \vdash E \quad \Gamma; \Delta \vdash^T T}{\Theta_1, T, \Theta_2; \Psi \vdash E}\ \mathrm{wk}_{\odot} \qquad \frac{\Theta; \Psi_1, \Psi_2 \vdash E \quad \Gamma; \Delta \vdash^T T}{\Theta; \Psi_1, T, \Psi_2 \vdash E}\ \mathrm{wk}_{\rhd}$$

These rules allow us to discard parts of a larger attack tree while constructing proof derivations, but this only works for attack trees not general ATLL formulas. Thus, with these rules ATLL becomes slightly affine. We can now use these rules to determine if $T_1 \multimap T_2$ holds, and if it does, then T_2 is a subattack tree of T_1.

3 Attack Trees with Costs

References

1. Carolyn Brown, Doug Gurr, and Valeria Paiva. A linear specification language for petri nets. *DAIMI Report Series*, 20(363), 1991.
2. Marcelo Fiore and Marco Devesas Campos. *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky: Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, chapter The Algebra of Directed Acyclic Graphs, pages 37–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
3. Luisa Francesco Albasini, Nicoletta Sabadini, and Robert F. C. Walters. The compositional construction of markov processes. *Applied Categorical Structures*, 19(1):425–437, 2010.
4. Ravi Jhawar, Barbara Kordy, Sjouke Mauw, SaÅ!'a RadomiroviÄ, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In Hannes Federrath and Dieter Gollmann, editors, *ICT Systems Security and Privacy Protection*, volume 455 of *IFIP Advances in Information and Communication Technology*, pages 339–353. Springer International Publishing, 2015.

$$\frac{}{\cdot;(E,c)\vdash_c E}\,\mathrm{id}_{\odot} \qquad \frac{}{(E,c);\cdot\vdash_c E}\,\mathrm{id}_{\triangleright}$$

$$\frac{\Theta;\Psi_1,(E_1,c_1),(E_2,c_2),\Psi_2\vdash_c E}{\Theta;\Psi_1,(E_2,c_2),(E_1,c_1),\Psi_2\vdash_c E}\,\mathrm{sym}_{\odot}$$

$$\frac{\Theta_1;\Psi_1\vdash_{c_1} E_1 \quad \Theta_2;\Psi_2\vdash_{c_2} E_2}{\Theta_1,\Theta_2;\Psi_1,\Psi_2\vdash_{\mathsf{op}_{\odot}(c_1,c_2)} E_1\odot_{\mathsf{op}_{\odot}}E_2}\,\odot_I$$

$$\frac{\Theta_1;\Psi_2\vdash_{\mathsf{op}_{\odot}(c_1,c_2)} E_1\odot_{\mathsf{op}_{\odot}}E_2 \quad \Theta_2;\Psi_1,(E_1,c_1),(E_2,c_2),\Psi_3\vdash_{c_3} E_3}{\Theta_1,\Theta_2;\Psi_1,\Psi_2,\Psi_3\vdash_{c_3} E_3}\,\odot_E$$

$$\frac{\Theta_1;\Psi_1\vdash_{c_1} E_1 \quad \Theta_2;\Psi_2\vdash_{c_2} E_2}{\Theta_1,\Theta_2;\Psi_1,\Psi_2\vdash_{\mathsf{op}_{\triangleright}(c_1,c_2)} E_1\triangleright_{\mathsf{op}_{\triangleright}}E_2}\,\triangleright_I$$

$$\frac{\Theta_2;\Psi_2\vdash_{\mathsf{op}_{\triangleright}(c_1,c_2)} E_1\triangleright_{\mathsf{op}_{\triangleright}}E_2 \quad \Theta_1,(E_1,c_1),(E_2,c_2),\Theta_3;\Psi_2\vdash_{c_3} E_3}{\Theta_1,\Theta_2,\Theta_3;\Psi_1,\Psi_2\vdash_{c_3} E_3}\,\triangleright_E$$

$$\frac{\Theta_1;\Psi_1\vdash_{c_1} E_1 \quad \Theta_2;\Psi_2\vdash_{c_2} E_2}{\Theta_1,\Theta_2;\Psi_1,\Psi_2\vdash_{\mathsf{op}_{\sqcup}(c_1,c_2)} E_1\sqcup_{\mathsf{op}_{\sqcup}}E_2}\,\sqcup \qquad \frac{\Theta;\Psi,(E_1,c_1)\vdash_{c_2} E_2 \quad \mathsf{rel}_{\multimap}(c_1,c_2)}{\Theta;\Psi\vdash_{c_2} E_1\multimap_{\mathsf{rel}_{\multimap}(c_1,-)}E_2}\,\multimap_I$$

$$\frac{\Theta_1;\Psi_1\vdash_{c_2} E_1\multimap_{\mathsf{rel}_{\multimap}(c_1,-)}E_2 \quad \Theta_2;\Psi_2\vdash_{c_1} E_1}{\Theta_1,\Theta_2;\Psi_1,\Psi_2\vdash_{c_2} E_2}\,\multimap_E$$

Fig. 4. Inference Rules for the Attack Tree Linear Logic (ATLL)

$$\frac{\Theta;\Psi\vdash_{c_1} T}{\Theta;\Psi\vdash_{c_1}(T\sqcup_{\mathsf{op}_{\sqcup}}T)\circ\!\!-\!\!\circ_{\mathsf{rel}_{\circ\!-\!\circ}(\mathsf{op}_{\sqcup}(c_1,c_1),-)}T}\,\mathrm{cont}_{\sqcup}$$

$$\frac{\Theta;\Psi\vdash_{c_1} T_1\sqcup_{\mathsf{op}_{\sqcup}}T_2}{\Theta;\Psi\vdash_{c}(T_1\sqcup_{\mathsf{op}_{\sqcup}}T_2)\circ\!\!-\!\!\circ_{\mathsf{rel}_{\circ\!-\!\circ}(c_1,-)}(T_2\sqcup_{\mathsf{op}_{\sqcup}}T_1)}\,\mathrm{sym}_{\sqcup}$$

$$\frac{\Theta;\Psi\vdash_{c_1}(T_1\sqcup_{\mathsf{op}_{\sqcup}}T_2)\sqcup_{\mathsf{op}_{\sqcup}}T_3}{\Theta;\Psi\vdash_{c}((T_1\sqcup_{\mathsf{op}_{\sqcup}}T_2)\sqcup_{\mathsf{op}_{\sqcup}}T_3)\circ\!\!-\!\!\circ_{\mathsf{rel}_{\circ\!-\!\circ}(c_1,-)}(T_1\sqcup_{\mathsf{op}_{\sqcup}}(T_2\sqcup_{\mathsf{op}_{\sqcup}}T_3))}\,\mathrm{assoc}_{\sqcup}$$

$$\frac{\Theta;\Psi\vdash_{c_1} T_1\odot_{\mathsf{op}_{\odot}}(T_2\triangleright_{\mathsf{op}_{\triangleright}}T_3)}{\Theta;\Psi\vdash_{c}(T_1\odot_{\mathsf{op}_{\odot}}(T_2\sqcup_{\mathsf{op}_{\sqcup}}T_3))\circ\!\!-\!\!\circ_{\mathsf{rel}_{\circ\!-\!\circ}(c_1,-)}((T_1\odot_{\mathsf{op}_{\odot}}T_2)\sqcup_{\mathsf{op}_{\sqcup}}(T_1\odot_{\mathsf{op}_{\odot}}T_3))}\,\mathrm{dis}_{\odot}$$

$$\frac{\Theta;\Psi\vdash_{c_1} T_1\triangleright_{\mathsf{op}_{\triangleright}}(T_2\sqcup_{\mathsf{op}_{\sqcup}}T_3)}{\Theta;\Psi\vdash_{c}(T_1\triangleright_{\mathsf{op}_{\triangleright}}(T_2\sqcup_{\mathsf{op}_{\sqcup}}T_3))\circ\!\!-\!\!\circ_{\mathsf{rel}_{\circ\!-\!\circ}(c_1,-)}((T_1\triangleright_{\mathsf{op}_{\triangleright}}T_2)\sqcup_{\mathsf{op}_{\sqcup}}(T_1\triangleright_{\mathsf{op}_{\triangleright}}T_3))}\,\mathrm{dis}_{\triangleright}$$

Fig. 5. Attack Tree Equivalence Axioms

5. Paul-André Melliès. Categorical semantics of linear logic. In Pierre-Louis Curien, Hugo Herbelin, Jean-Louis Krivine, and Paul-André Melliès, editors, *Interactive models of computation and program behaviour*. Panoramas et Synthèses 27, Société Mathématique de France, 2009.
6. A Tzouvaras. The linear logic of multisets. *Logic Journal of IGPL*, 6(6):901–916, 1998.

Appendix

A Symmetric Monoidal Categories

This appendix provides the definitions of both categories in general, and, in particular, symmetric monoidal closed categories. We begin with the definition of a category:

Definition 3. *A **category**, $\mathcal{C}$, consists of the following data:*

- *A set of objects $\mathcal{C}_0$, each denoted by A, B, C, etc.*
- *A set of morphisms $\mathcal{C}_1$, each denoted by f, g, h, etc.*
- *Two functions src, the source of a morphism, and tar, the target of a morphism, from morphisms to objects. If $\mathsf{src}(f) = A$ and $\mathsf{tar}(f) = B$, then we write $f : A \to B$.*
- *Given two morphisms $f : A \to B$ and $g : B \to C$, then the morphism $f; g : A \to C$, called the composition of f and g, must exist.*
- *For every object $A \in \mathcal{C}_0$, the there must exist a morphism $\mathsf{id}_A : A \to A$ called the identity morphism on A.*
- *The following axioms must hold:*
  - *(Identities) For any $f : A \to B$, $f; \mathsf{id}_B = f = \mathsf{id}_A; f$.*
  - *(Associativity) For any $f : A \to B$, $g : B \to C$, and $h : C \to D$, $(f; g); h = f; (g; h)$.*

Categories are by definition very abstract, and it is due to this that makes them so applicable. The usual example of a category is the category whose objects are all sets, and whose morphisms are set-theoretic functions. Clearly, composition and identities exist, and satisfy the axioms of a category. A second example is preordered sets, $(A, \leq)$, where the objects are elements of A and a morphism $f : a \to b$ for elements $a, b \in A$ exists iff $a \leq b$. Reflexivity yields identities, and transitivity yields composition.

Symmetric monoidal categories pair categories with a commutative monoid like structure called the tensor product.

Definition 4. *A **symmetric monoidal category (SMC)** is a category, $\mathcal{M}$, with the following data:*

- *An object I of $\mathcal{M}$,*
- *A bi-functor $\otimes : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$,*

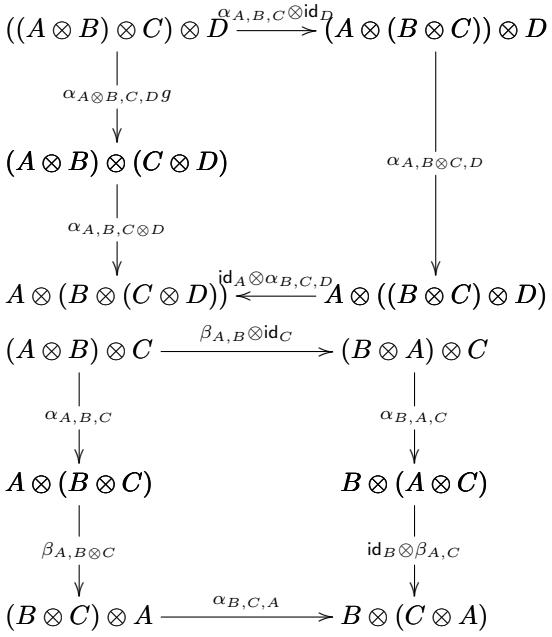
– *The following natural isomorphisms:*

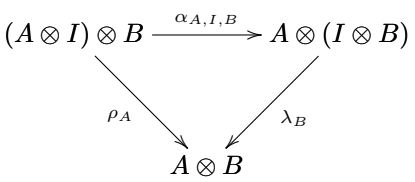
$$\lambda_A : I \otimes A \to A$$
$$\rho_A : A \otimes I \to A$$
$$\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$$

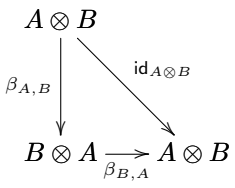
– *A symmetry natural transformation:*

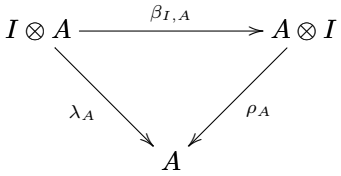
$$\beta_{A,B} : A \otimes B \to B \otimes A$$

– *Subject to the following coherence diagrams:*









Monoidal categories posses additional structure, and hence, ordinary functors are not enough, thus, the notion must also be extended.

Definition 5. *Suppose we are given two monoidal categories* $(\mathcal{M}_1, \top_1, \otimes_1, \alpha_1, \lambda_1, \rho_1)$ *and* $(\mathcal{M}_2, \top_2, \otimes_2, \alpha_2, \lambda_2, \rho_2)$*. Then a* ***monoidal functor*** *is a functor* $F : \mathcal{M}_1 \longrightarrow \mathcal{M}_2$*, a map* $m_{\top_1} : \top_2 \longrightarrow F\top_1$ *and a natural transformation* $m_{A,B} : FA \otimes_2 FB \longrightarrow F(A \otimes_1 B)$ *subject to the following coherence conditions:*

$$\begin{array}{ccc}
(FA \otimes_2 FB) \otimes_2 FC & \xrightarrow{\alpha_{2\,FA,FB,FC}} & FA \otimes_2 (FB \otimes_2 FC) \\
\downarrow{\scriptstyle m_{A,B} \otimes \mathsf{id}_{FC}} & & \downarrow{\scriptstyle \mathsf{id}_{FA} \otimes m_{B,C}} \\
F(A \otimes_1 B) \otimes_2 FC & & FA \otimes_2 F(B \otimes_1 C) \\
\downarrow{\scriptstyle m_{A \otimes_1 B, C}} & & \downarrow{\scriptstyle m_{A, B \otimes_1 C}} \\
F((A \otimes_1 B) \otimes_1 C) & \xrightarrow{F\alpha_{1\,A,B,C}} & F(A \otimes_1 (B \otimes_1 C))
\end{array}$$

$$\begin{array}{ccc}
\top_2 \otimes_2 FA & \xrightarrow{\lambda_{2\,FA}} & FA \\
\downarrow{\scriptstyle m_{\top_1} \otimes \mathsf{id}_{FA}} & & \uparrow{\scriptstyle F\lambda_{1\,A}} \\
F\top_1 \otimes_2 FA & \xrightarrow{m_{\top_1, A}} & F(\top_1 \otimes_1 A)
\end{array}
\qquad
\begin{array}{ccc}
FA \otimes_2 \top_2 & \xrightarrow{\rho_{2\,FA}} & FA \\
\downarrow{\scriptstyle \mathsf{id}_{FA} \otimes m_{\top_1}} & & \uparrow{\scriptstyle F\rho_{1\,A}} \\
FA \otimes_2 F\top_1 & \xrightarrow{m_{A, \top_1}} & F(A \otimes_1 \top_1)
\end{array}$$

B Source Sink Graphs are Symmetric Monoidal

In this appendix I show that the category of source-sink graphs defined by Jhawar et al. [4] is symmetric monoidal. First, recall the definition of source-sink graphs and their homomorphisms.

Definition 6. *A* ***source-sink graph*** *over* B *is a tuple* $G = (V, E, s, z)$*, where* V *is the set of vertices,* E *is a multiset of labeled edges with support* $E^* \subseteq V \times \mathsf{B} \times V$*,* $s \in V$ *is the unique start,* $z \in V$ *is the unique sink, and* $s \neq z$*.*

Suppose $G = (V, E, s, z)$ *and* $G' = (V', E', s', z')$*. Then a* ***morphism between source-sink graphs****,* $f : G \to G'$*, is a graph homomorphism such that* $f(s) = s'$ *and* $f(z) = z'$*.*

Suppose $G = (V, E, s, z)$ and $G' = (V', E', s', z')$ are two source-sink graphs. Then given the above definition it is possible to define sequential and non-communicating parallel composition of source-sink graphs where I denote disjoint union of sets by $+$ (p 7. [4]):

Sequential Composition :
$G \triangleright G' = ((V \setminus \{z\}) + V', E^{[s'/z]} + E', s, z')$

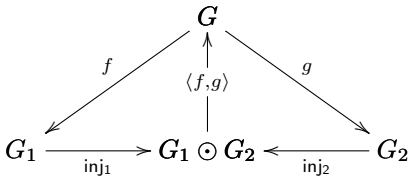
Parallel Composition :
$G \odot G' = ((V \setminus \{s, z\}) + V', E^{[s'/s, z'/z]} + E', s', z')$

It is easy to see that we can define a category of source-sink graphs and their homomorphisms. Furthermore, it is a symmetric monoidal category were parallel composition is the symmetric tensor product. It is well-known that any category with co-products is symmetric monoidal where the co-product is the tensor product.

I show here that parallel composition defines a co-product. This requires the definition of the following morphisms:

$$\begin{aligned}
&\mathsf{inj}_1 : G_1 \to G_1 \odot G_2 \\
&\mathsf{inj}_2 : G_2 \to G_1 \odot G_2 \\
&\langle f, g \rangle : G_1 \odot G_2 \to G
\end{aligned}$$

In the above $f : G_1 \to G$ and $g : G_2 \to G$ are two source-sink graph homomorphisms. Furthermore, the following diagram must commute:



Suppose $G_1 = (V_1, E_1, s_1, z_1)$, $G_2 = (V_2, E_2, s_2, z_2)$, and $G = (V, E, s, z)$ are source-sink graphs, and $f : G_1 \to G$ and $g : G_2 \to G$ are source-sink graph morphisms – note that $f(s_1) = g(s_2) = s$ and $f(z_1) = g(z_2) = z$ by definition. Then we define the required co-product morphisms as follows:

$$\begin{aligned}
&\mathsf{inj}_1 : V_1 \to (V_1 \setminus \{s_1, z_1\}) + V_2 \\
&\mathsf{inj}_1(s_1) = s_2 \\
&\mathsf{inj}_1(z_1) = z_2 \\
&\mathsf{inj}_1(v) = v, \text{ otherwise}
\end{aligned}$$

$$\begin{aligned}
&\mathsf{inj}_2 : V_2 \to (V_1 \setminus \{s_1, z_1\}) + V_2 \\
&\mathsf{inj}_2(v) = v
\end{aligned}$$

$$\begin{aligned}
&\langle f, g \rangle : (V_1 \setminus \{s_1, z_1\}) + V_2 \to V \\
&\langle f, g \rangle(v) = f(v), \text{ where } v \in V_1 \\
&\langle f, g \rangle(v) = g(v), \text{ where } v \in V_2
\end{aligned}$$

It is easy to see that these define graph homomorphisms. All that is left to show is that the diagram from above commutes:

$$\begin{aligned}
(\mathsf{inj}_1; \langle f, g \rangle)(s_1) &= \langle f, g \rangle(\mathsf{inj}_1(s_1)) \\
&= g(s_2) \\
&= s \\
&= f(s_1)
\end{aligned}$$

$$\begin{aligned}
(\mathsf{inj}_1; \langle f, g \rangle)(z_1) &= \langle f, g \rangle(\mathsf{inj}_1(z_1)) \\
&= g(z_2) \\
&= z \\
&= f(z_1)
\end{aligned}$$

Now for any $v \in V_1$ we have the following:

$$\begin{aligned}(\mathsf{inj_1}; \langle f, g \rangle)(v) &= \langle f, g \rangle(\mathsf{inj_1}(v)) \\ &= f(v)\end{aligned}$$

The equation for $\mathsf{inj_2}$ is trivial, because $\mathsf{inj_2}$ is the identity.